(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（A）**　(11)特許出願公開番号

特開平6－19797

(43)公開日　平成６年(1994)１月28日

(51)Int.Cl.[5]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　12/12		Ｅ　7608－5Ｂ		

審査請求　未請求　請求項の数３(全 10 頁)

(21)出願番号　特願平5－123090

(22)出願日　平成５年(1993)４月28日

(31)優先権主張番号　８７５，３５７

(32)優先日　1992年４月29日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　591064003
サン・マイクロシステムズ・インコーポレーテッド
SUN MICROSYSTEMS, INCORPORATED
アメリカ合衆国　94043　カリフォルニア州・マウンテンビュー・ガルシア　アヴェニュ・2550

(72)発明者　アダム・マラミイ
アメリカ合衆国　01890　マサチューセッツ州・ウィンチェスター・ワイルドウッド　ストリート・39

(74)代理人　弁理士　山川　政樹

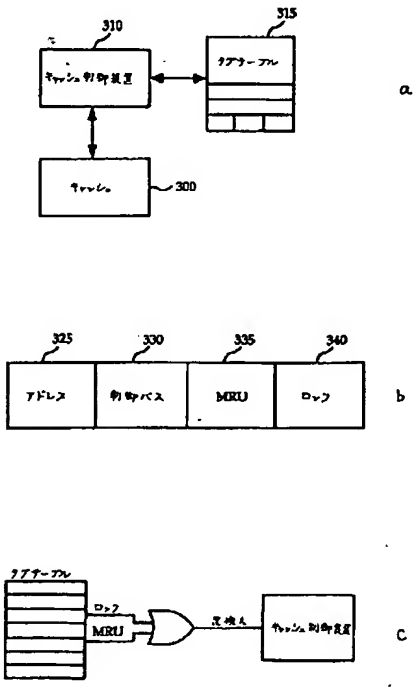
最終頁に続く

(54)【発明の名称】　キャッシュ内の記憶されている行を保護する装置及び方法

(57)【要約】

【目的】　主メモリと、より高速のキャッシュメモリとを有するメモリシステムにおいて、ロッキング機能を伴うキャッシュメモリ置換え方式を提供する。

【構成】　タグテーブル中に、キャッシュのそれぞれの行と関連するロッキングビットが供給される。それらのロッキングビットは実行中のアプリケーションプログラム／プロセスによりセット、リセットされるのが好ましく、置換えるべきキャッシュの行を確定するために、キャッシュ制御装置によりキャッシュ置換えビットと関連して利用される。ロッキングビットは、プロセスがロックビットをリセットする時点までキャッシュ内のデータの行を「ロック」する。プロセスがロックビットの状態を制御するように規定することにより、いくつかのメモリ記憶場所の使用頻度に関してプロセスが含んでいる知能と知識を利用して、さらに効率良いキャッシュを提供できる。



1

【特許請求の範囲】

【請求項1】　中央処理装置（ＣＰＵ）を含むマスター装置と、主メモリ及びより高速のキャッシュメモリから構成されるメモリシステムとを具備し、メモリに対してアクセスの要求を発行したマスター装置によって高速アクセスできるように主メモリの行のサブセットがキャッシュメモリに記憶されるようなコンピュータシステムにあって、キャッシュメモリの主メモリから選択された行を保護する装置において、

キャッシュに記憶されている主メモリの行を識別し、キャッシュからの行を供給することによりメモリに対するアクセスを実行するか否かを判定するために比較される複数のタグビットと、少なくとも１つのロックビットとをキャッシュの行ごとに、含むタグテーブルと；タグテーブルに含まれているキャッシュの行ごとのロックビットの状態を制御する手段と；キャッシュに記憶されている１つのメモリ行を異なるメモリ行と置換えるが、タグテーブルにおける対応するロックビットがセットされている場合には、メモリ行を置換えるのを禁止される置換え手段とを具備し、

タグテーブル中の対応するロックビットをセットすることにより、キャッシュ置換えアルゴリズムとは無関係に、キャッシュに記憶されているメモリ行はキャッシュ内で保護される装置。

【請求項2】　中央処理装置（ＣＰＵ）を含むマスター装置と、主メモリ、より高速のキャッシュメモリ及びキャッシュメモリと関連するタグテーブルから構成されるメモリシステムとを具備し、メモリに対するアクセスの要求を発行したマスター装置によって高速アクセスできるように主メモリの行のサブセットがキャッシュメモリに記憶されるコンピュータシステムで、かつ前記タグテーブルは複数のタグビットを含み、前記タグビットはキャッシュに記憶されている主メモリの行を識別し、そのタグビットが、キャッシュからの行を供給することによりメモリに対するアクセスを実行するか否かを判定するために比較されるようなコンピュータシステムでキャッシュメモリの主メモリが選択された行を保護する方法において、

タグテーブル中のキャッシュのそれぞれの行と関連する少なくとも１つのロックビットを提供する過程と；タグテーブルに含まれているキャッシュの行ごとのロックビットの状態を制御する過程と；セットされている関連ロックビットを有するキャッシュの行は置換えられないように、キャッシュに記憶されている１つのメモリ行を異なるメモリ行と置換える過程とから成り、

タグテーブル中の対応するロックビットをセットすることにより、キャッシュ置換えアルゴリズムとは無関係に、キャッシュに記憶されているメモリ行はキャッシュ内で保護されるような方法。

【請求項3】　中央処理装置（ＣＰＵ）を含むマスター装置と、主メモリ及びより高速のキャッシュメモリから構成されるメモリシステムとを具備し、メモリに対するアクセスの要求を発行したマスター装置によって高速アクセスするために主メモリの行のサブセットがキャッシュメモリに記憶されるようなコンピュータシステムのキャッシュメモリで主メモリの選択された行を保護する装置において、

キャッシュに記憶されている主メモリの行を識別し、キャッシュからの行を供給することによりメモリに対するアクセスを実行するか否かを判定するために比較される複数のタグビットと、少なくとも１つの置換えビットと、少なくとも１つのロックビットとをキャッシュの行ごとに含むタグテーブルと；タグテーブルに含まれているロックビットをセット／リセットするための指令を発行するオペレーティングシステムと；キャッシュに記憶されている１つのメモリ行を異なるメモリ行と置換えるための置換えアルゴリズムを実行し且つタグテーブルを更新するが、タグテーブル中の対応するロックビットがセットされている場合には、キャッシュ内のメモリ行を置換えることを禁止するキャッシュ及びタグテーブルの内容を制御するキャッシュ制御装置とを具備し、

タグテーブル中の対応するロックビットをセットすることにより、キャッシュ置換えアルゴリズムとは無関係に、キャッシュに記憶されているメモリ行がキャッシュ内で保護される装置。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、コンピュータのキャッシュメモリ装置の分野に関する。さらに詳細には、本発明は、プログラムがキャッシュにとどまるべきメモリのページ又はブロックを指定できるようにデータをキャッシュメモリに「ロックする」方法及び装置に関する。

【０００２】

【従来の技術】コンピュータプロセッサのスループットを向上させるための単純な方法は、プロセッサを駆動するクロックの周波数を増加させるというものである。ところが、プロセッサのクロック周波数が増すと、プロセッサはプロセッサの要求に主メモリが応答しうる速度を上回る速度で動作し始めてしまう場合がある。従って、主メモリが応答するまで、プロセッサは待機せざるをえないことがある。この主メモリ待ち時間を緩和するために、キャッシュメモリが構築されたのである。

【０００３】キャッシュメモリとは、プロセッサに密接して結合する小容量の高速メモリである。キャッシュメモリは主メモリの記憶場所のサブセットを複製するために使用される。プロセッサがメモリのデータを必要としたときには、プロセッサは、まず、高速キャッシュメモリを探索する。キャッシュメモリの中でそのデータが見つかれば（「ヒット」として知られている）、データをキャッシュメモリから検索し、実行を再開する。キャッ

シュメモリでデータを見つけられない（「ミス」として知られている）場合には、プロセッサは続いてより低速の主メモリを探索する。

【0004】たとえば、特定のプログラムが主メモリの特定のデータテーブルを頻繁に参照する場合、そのデータテーブルのコピーを高速キャッシュメモリに導入することが望ましいであろう。データテーブルのコピーをキャッシュメモリに保持しておけば、プロセッサがそのデータテーブルに含まれるデータを要求するたびに、そのデータは迅速に検索される。

【0005】通常、キャッシュメモリは主メモリの小形のサブセットのみを記憶している。キャッシュメモリの記憶場所を充填するたびに、キャッシュメモリは現在記憶されているデータの一部を廃棄しなければならない。多くの場合に、将来、どのキャッシュメモリ記憶場所が必要になるかはわかっていないので、どのキャッシュメモリ記憶場所を廃棄すべきかを確定するのは難しいタスクである。主メモリのどの記憶場所を高速キャッシュメモリで複製するかを判定するのを助けるために、様々なヒューリスティックが開発されている。

【0006】図1を参照すると、従来のキャッシュメモリシステムの高レベルブロック線図が示されている。主メモリ10と、キャッシュメモリシステム12と、プロセッサ14はバス16に結合している。プロセッサはメモリ要求をキャッシュメモリシステム12へ発行する。キャッシュメモリ15で情報を利用可能であれば、要求された情報を専用回線18を介して直ちにプロセッサ14へ送り出す。情報がキャッシュメモリ15に記憶されていない場合には、要求を低速の主メモリ10へ送り、主メモリは要求された情報をバス16を介してプロセッサ14へ送る。

【0007】物理主メモリアドレスをキャッシュメモリの記憶場所にマッピングする方法は数多くある。それらの方法の中には、フルアソシエイティブ方式、直接マッピング方式及びセットアソシエイティブ方式がある。フルアソシエイティブキャッシュシステムでは、主メモリのいずれかのブロックをいずれかのキャッシュメモリ行で表示することができる。直接マッピング方式のシステムにおいては、主メモリの各ブロックを唯一つの特定のキャッシュメモリ記憶場所で表示することができる。セットアソシエイティブシステムでは、主メモリの各ブロックを同一のセット番号を有するキャッシュメモリ行に導入することのみ可能である。キャッシュメモリマッピングシステムの詳細については、Hennessy, Pattersonの「Cmputer Architecture:A Quantitative Approach」（Morgan Kaufman Press, 1990年刊）の408～410ページを参照。

【0008】キャッシュメモリの動作を制御するために、キャッシュ制御装置（図1の17）と呼ばれる専用制御論理が設けられている。キャッシュ制御装置の中には1つのテーブルがある。TAGテーブルは、主メモリの物理アドレスをキャッシュメモリセット及び行アドレスにマッピングするために使用する情報を記憶するために使用される。詳細にいえば、TAGテーブルはキャッシュメモリ行ごとのブロックアドレスと関連制御ビットを記憶しているということになる。ブロックアドレスは、キャッシュメモリ行で現在表示されている物理主メモリブロックアドレスを表わす。制御ビットは、キャッシュメモリ行が有効データを有するか否かなどの情報を記憶している。加えて、テーブルはキャッシュ置換えアルゴリズムを実現するために利用するデータを記憶している。データテーブルはキャッシュメモリの編成に一致するように分割されている。

【0009】1つのキャッシュメモリセットの中の全ての行が一杯になり、新たなメモリのブロックをキャッシュメモリに導入すべきである場合には、キャッシュ制御装置はキャッシュメモリの一部の内容を廃棄して、それを主メモリからの新たなデータと置換えなければならない。廃棄するキャッシュメモリ行の内容は近い将来に必要とされないものであるのが好ましい。ところが、キャッシュ制御装置はどのキャッシュメモリ行を廃棄すべきかを予測することしかできない。先に簡単に述べた通り、できる限り効率良く予測するために、いくつかのキャッシュ置換えヒューリスティックが開発されている。現在使用されているキャッシュ置換えヒューリスティックはラウンドロビン、ランダム、最低使用頻度（LRU）、疑似最低使用頻度などである。これらのヒューリスティックは、キャッシュメモリの過去の性能のみを見ることにより、どのキャッシュメモリ記憶場所を置換えるべきかを確定する。

【0010】ラウンドロビン置換えヒューリスティックは単純にキャッシュメモリ行を逐次順に置換えてゆく。最後のキャッシュメモリ行に到達すると、制御装置は第1のキャッシュメモリ行に戻って始める。

【0011】最低使用頻度（LRU）置換え方式はキャッシュ制御装置にさらに高い知能を要求する。LRUヒューリスティックにおいては、最近の時点でキャッシュメモリの1つの行をアクセスした場合、その行が近い将来に再びアクセスされる確率は高いということを仮定する。この仮定に基づけば、「最も遠い時点で使用された」キャッシュメモリ行をキャッシュ制御装置により置換えるべきであるということになる。LRUヒューリスティックを実現するために、キャッシュメモリのそれぞれの行で「ヒット」が起こるたびに、キャッシュ制御装置はタイムカウンタによってそのキャッシュメモリ行をマークしなければならない。キャッシュ制御装置がキャッシュメモリ行を置換えることを強いられると、キャッシュ制御装置はそのキャッシュメモリ行を最も古いタイムカウンタ値と置換える。このようにして、「最も遠い

時点で使用された」キャッシュメモリ行を置換える。

【0012】LRUヒューリスティックは相対的に効率が良いが、欠点もある。LRU置換え方式に関わる1つの問題は、この方式が貴重な高速キャッシュメモリを浪費することである。キャッシュヒットが起こるたびに、キャッシュ制御装置はそのキャッシュメモリ行と関連する記憶場所にタイムカウンタ値を導入しなければならない。LRU置換え方式に関わるもう1つの問題は、実現に際して複雑な論理を要求することである。置換えを実行しなければならないときには、キャッシュ制御装置は全てのキャッシュメモリ行のタイムカウンタ値を比較しなければならず、この手続きで貴重な時間を浪費してしまう。これらの要因が取込まれると、LRU方式の効率は幾分か低下する。

【0013】疑似最低使用頻度（PLRU）置換え方式は、それほど複雑な論理を必要とせず且つ実現に際してごく高速のキャッシュメモリを要求しないという点を除いて、LRU置換え方式にやや似ている。しかしながら、PLRU方式は動作をスピードアップするために手っ取り早い方法を採用するので、使用頻度が最低であるキャッシュメモリ記憶場所が置換える記憶場所であるとは限らない。PLRU置換え方式においては、各キャッシュメモリ行に、TAGテーブルに記憶されるMRU（すなわち、最高使用頻度）ビットを割当てる。キャッシュメモリ行ごとのMRUビットは、そのキャッシュメモリ行で「ヒット」が起こるたびに「1」にセットされる。すなわち、MRUビットの「1」は、そのキャッシュメモリ行が最近の時点で使用されたことを指示する。キャッシュ制御装置が1つのキャッシュメモリ行を置換えることを強いられたときには、キャッシュ制御装置はキャッシュメモリ行ごとにMRUビットを「0」を求めて検査する。ある特定のキャッシュメモリ行のMRUビットが「1」にセットされていれば、そのキャッシュメモリ行は最近の時点で使用された行であるので、キャッシュ制御装置はそのキャッシュメモリ行を置換えない。キャッシュ制御装置が「0」にセットされているMRUビットを伴うメモリ行を見出すと、そのメモリ行が置換えられることになり、そのキャッシュメモリ行と関連するMRUビットを「1」にセットすることになる。

【0014】全てのキャッシュメモリ行が「1」にセットされた場合に、問題が起こるおそれが生じるであろう。この事態が起こると、全ての行は置換えのために利用できなくなるので、そこでデッドロックを発生させる。この種のデッドロックを阻止するために、オーバフロー状況が起こりうると検出されたときには、アクセス中のMRUビットを除いてTAGの全MRUビットをクリアする。キャッシュがセットアソシエイティブである場合には、オーバフロー状況が起こりうると検出されたとき、そのセットの全てのMRUビットは「1」にセットされているので、アクセス中のMRUビットを除いてそのセットに関わるTAGアレイ中の全MRUビットをクリアする。

【0015】PLRU方式は1つの例を利用することにより最もわかりやすく説明される。図2を参照すると、4つのキャッシュ行を利用できるキャッシュ環境におけるPLRU置換え方式の1例が示されている。ステップ1では、近い時点でどのキャッシュ行も使用されておらず、全てのキャッシュ行を自在に置換えられることを指示するように、全てのMRUビットをクリアする。ステップ2では、行3のデータについてキャッシュヒットが起こる。キャッシュ制御装置は行3に関わるMRUビットを「1」にセットさせ、行3のデータが近い時点で使用されたことを指示する。キャッシュ行0，1及び2は依然として利用可能である。ステップ3では、行1のデータについてキャッシュヒットが起こる。キャッシュ制御装置は行1に関わるMRUビットを「1」にセットさせ、行1のデータが近い時点で使用されたことを指示する。ステップ4では、行0のデータについてキャッシュヒットが起こる。キャッシュ制御装置は同様に行0に関わるMRUビットを「1」にセットさせ、行0のデータが近い時点で使用されたことを指示する。その時点で、近い時点で使用されたとマークされていないキャッシュ行は行2のみである。ステップ5では、行2のデータについてキャッシュヒットが起こる。行2に関わるMRUビットが「1」にセットされれば、全てのMRUビットは「1」にセットされ、（1111）、置換えのために利用できるキャッシュ行はなくなってしまうであろう。これがキャッシュデッドロックのケースであると考えられる。そのようにする代わりに、キャッシュ制御装置は全てのMRUビットをクリアさせ、行2に関わるMRUビットを「1」にセットする。その時点で、置換えのために利用できる行は行0，1及び3である。全MRUビットのクリアという動作の結果、キャッシュの履歴の一部は失われるが、キャッシュのデッドロックを回避するためには、この動作は必要である。その後、キャッシュ動作は以前と同様に続いてゆく。

【0016】ところが、キャッシュメモリの将来の利用状況に関わる情報が幾分かでもわかれば、上述のヒューリスティックを改善することができるのである。たとえば、近い将来、ある1つのキャッシュメモリ記憶場所が使用されるということがわかれば、そのキャッシュメモリ記憶場所を置換えないでおくのが最良であろう。先に挙げた例では、プログラムはデータテーブル中のデータを繰返しアクセスするであろうということがわかっていた。その場合、データテーブルをキャッシュメモリに導入すると、そのキャッシュメモリ記憶場所を置換えることができないように、そのキャッシュメモリ記憶場所を「ロック」可能とすることが有利であろう。これを実行したならば、プログラムがその後にデータテーブルから情報を要求するたびに、そのデータは常にキャッシュメ

モリで見つかるであろう。従って、データテーブル中のデータは、低速の主メモリから検索する必要なく、キャッシュメモリから迅速に取出されるであろう。

【0017】

【発明が解決しようとする課題】従って、本発明の目的は、キャッシュメモリが一杯になったときにキャッシュメモリ記憶場所を置換える効率良い方法を提供することである。本発明の別の目的は、プログラムにいくつかのキャッシュメモリ記憶場所を、それらの記憶場所が置換えられないようにキャッシュメモリをロックさせる方法及び装置を提供することである。本発明の別の目的は、ユーザーに全てのキャッシュメモリ記憶場所をロックさせないことにより、ユーザーがキャッシュメモリの「デッドロック」を発生させるのを阻止することである。

【0018】

【課題を解決するための手段】上記の目的及びその他の目的は、本発明の独自の方法と装置により達成される。本発明の方法及び装置は、ロッキングビットを利用するキャッシュメモリ置換え方式から成る。それらのロッキングビットは実行中のアプリケーションプログラム／プロセスによりセット、リセットされるのが好ましく、キャッシュ制御装置により、置換えるべきキャッシュの行を確定するためにキャッシュ置換えビットと関連して利用される。ロッキングビットは、プロセスがロックビットをリセットする時点まで、キャッシュ中のデータの行を「ロック」する。プロセスがロックビットの状態を制御すると規定するならば、いくつかの記憶場所の使用頻度に関してプロセスが含んでいる知能と知識を利用して、さらに効率の良いキャッシュを提供することができる。本発明の目的、特徴及び利点は以下の詳細な説明から当業者には明白になるであろう。

【0019】

【実施例】最低使用頻度置換えアルゴリズムを実現するキャッシュに、キャッシュ内のいくつかのメモリ記憶場所をロックする能力を与える。キャッシュのメモリ記憶場所がロックされると、その場所に記憶されている情報は、ロックが解除され、キャッシュ置換えアルゴリズムがキャッシュのその行を置換えるべきであると判定するまでキャッシュ内にとどまる。

【0020】タグテーブルには、追加ビットとして、キャッシュメモリのそれぞれの行と関連するロックビットが含まれている。その特定のキャッシュメモリ記憶場所をアクセスするプロセスによって、このビットをセットできるのが好ましい。利点は、知能が追加され且つキャッシュをアクセスしているアプリケーションプログラム又はプロセスにより既存の知識が与えられていることである。アプリケーションプログラムは、プログラム実行中のいくつかの変数又はメモリのアクセスの頻度に関する既存の知識を有する。この知識は、置換えアルゴリズムを実現するキャッシュ制御装置には容易には明らかにならない。従って、キャッシュ制御装置、あるいはキャッシュ置換えアルゴリズムの複雑さを過度に増さずに、キャッシュ置換えアルゴリズムの知能の向上が得られる。

【0021】以下の説明中、本発明を完全に理解させるために、特定の用語を挙げるが、本発明を実施するに際してそれらの特定の詳細な事項が要求されないことは当業者には明白であろう。また、場合によっては、本発明を無用にわかりにくくしないために、周知の回路や装置をブロック線図の形で示すことがある。特定すれば、本発明はセットアソシエイティブマッピングシステムと、疑似最低使用頻度置換えアルゴリズムとを使用して実現されている。しかしながら、当業者には明白であるように、本発明のキャッシュシステムはセットアソシエイティブマッピングを伴うキャッシュメモリシステム又は疑似最低使用頻度置換えアルゴリズムには限定されない。図3を参照すると、セットアソシエイティブキャッシュメモリの1例のブロック線図が示されている。図示した例のセットアソシエイティブキャッシュメモリシステムでは、キャッシュメモリ「セット」は64あり、それぞれのセットに0～63のラベルが付されている。キャッシュメモリの各セットはキャッシュメモリの4つの「行」を含む。各セットのキャッシュメモリのそれぞれの行には0～3のラベルが付されている。それぞれのキャッシュメモリ行は主メモリの1つの「ブロック」全体を記憶することができる。

【0022】キャッシュメモリと同様に、主メモリも多数のセットに分割されている。主メモリが分割されるセットの数はキャッシュメモリ中のセットの数と等しい。たとえば、図3に示すように、主メモリは64のセットに分割されている。主メモリはブロックアドレスの上位ビットに従って分割される。すなわち、初めのn個のブロックはセット0に属し、次のn個のブロックはセット1に属する。続くブロックも同様である。0で終わる全てのブロックアドレスがセット0に属し、1で終わる全てのブロックアドレスはセット1に属するように、ブロックアドレスの下位ビットを使用しても全く同じように容易にセットを分割できるであろうということは明白である。たとえば、セット0はブロック0，N，2N，・・・，61N，62N，63Nを含み、セット1はブロック1，N＋1，2N＋1，・・・，61N＋1，62N＋1，63N＋1を含むということになる。

【0023】主メモリのセットはキャッシュメモリのセットより相当に大きい。主メモリの各セットは多数のメモリブロックにさらに分割されている。主メモリの各ブロックは同一のセット番号をもつキャッシュメモリのセットでのみ複製可能である。例を挙げると、セット0のブロック3はキャッシュメモリのセット0でのみ複製可能であり、セット1のブロックn＋1はキャッシュメモリのセット1でのみ複製可能である。

【0024】先に述べた通り、キャッシュメモリの各セットはキャッシュメモリの多数の「行」から構成されている。キャッシュメモリの「行」のサイズは主メモリの「ブロック」と等しく、主メモリのブロックの複製を記憶するために使用される。本質的には、「行」はキャッシュメモリにのみあり、ブロックは主メモリにのみあるという点を除いて、キャッシュメモリの行と主メモリのブロックとは同じである。

【0025】本発明のキャッシュシステムのロッキングメカニズムの概念を図4を参照して説明する。図4aは、最も近い時点でアクセスされたアドレスのメモリ内容を記憶するキャッシュ300を示す。キャッシュ制御装置310はキャッシュ300に対するアクセスを制御し、キャッシュを更新するために、キャッシュ置換えアルゴリズムを実行する。タグテーブル315はメモリに関する情報、すなわち、キャッシュに記憶されているデータのタグアドレスと、制御ビットとを含む。図4bを参照すると、タグテーブルのエントリの1例が示されている。キャッシュの行ごとに、1つのタグテーブルエントリが設けられている。アドレス325及び制御ビット330に加えて、各エントリに、その特定の行でキャッシュがアクセスされたときにセットされるビットMRU335が与えられている。キャッシュ制御装置が実行する置換えアルゴリズムで、これを利用するのである。さらに、キャッシュの行が置換えられるのを阻止するためのロックビット340がある。このロックビットはキャッシュをアクセスするプロセッサプログラムによりセット可能であり、また、同様に、その情報に対する繰返しアクセスが不要になり、キャッシュの行を置換えられるようになったときに、そのプログラムによりセット可能である。図4cは、概念をどのように実現するかを目で見てわかるように示している。キャッシュ制御装置がキャッシュの行を置換えることを要求されると、キャッシュ制御装置はMRUを読取り且つデータをロックするためにタグテーブルをアクセスする。そこで、ロックデータとMRUデータとを論理的に論理和演算して、キャッシュのその特定の行を置換えることができるか否かを示す置換えビットを得る。この論理和演算機能はキャッシュ制御装置自体により実行されても良いし、あるいは、外部論理により実行されても良い。論理和演算機能の結果は合成マスクとして知られている。合成マスクの中の得られた置換えビットがセットされていれば、キャッシュの行は異なるメモリ記憶場所による置換えのために除去されない。従って、MRUビットの値にかかわらず、データをキャッシュに確実に維持しておくために、ロックビットをセットすることができる。

【0026】年　月　日に出願された名称「Cache Set Tag Array」による同時係属米国特許出願第＿＿＿号に記載してある通り、タグテーブルは2つの別個のタグテーブルとして実現されるのが好ましい。このことを図5に示す。第1のテーブルPTAG400はアドレス情報と、制御ビットとを含む。アドレスは、キャッシュメモリ行で現在表示されている物理主メモリブロックアドレスである。制御ビットは、キャッシュメモリ行が有効データを含んでいるか否かを指示する有効ビットを含む。加えて、第2のテーブルSTAG410も設けられている。STAGはキャッシュメモリの行ごとのMRUビットと、ロックビットとを含む。先に述べた通り、MRUビットは疑似最低使用頻度置換え方式を実現するために使用される。

【0027】全ての行に関わる全ての合成ビットがセットされ、キャッシュデッドロックが起こるような状態に合成マスクが陥ることが決してないよう保証するために、キャッシュ制御装置は合成マスクの状態を監視する。加えて、全てのキャッシュメモリ行がユーザーによりロックされるのを阻止するために、セット状態のロックビットの数を監視すると共に、所定数のロックビットがセットされていた場合にアプリケーションプログラムによる追加のロック要求を抑止するためのメカニズムが設けられている。メカニズムはキャッシュ制御装置、プログラム／プロセス又はコンパイラに設けられれば良い。あるいは、全てのキャッシュ行のロックを回避するために、ロックビットが決してセットされないようにキャッシュメモリ行0を制御するのが好ましい。これにより、問題を単純に且つオーバヘッドも少なく解決でき、プログラマーの誤りに起因するデッドロックは回避される。

【0028】図6を参照すると、本発明の置換え方式の利用例が挙げられている。ステップ1の初期開始点では、全てのMRUビットとロックビットをクリアする。ステップ2では、行3のデータについてキャッシュヒットが起こる。キャッシュ制御装置はキャッシュメモリ行3に関わるMRUビットを、行3のデータが近い時点で使用されたことを指示する「1」にセットさせる。キャッシュ行0，1及び2は依然として利用可能である。次のステップ3では、ユーザープログラムは行2にあるデータをロックする。そこで、キャッシュ制御装置はキャッシュメモリ行2に関わるロックビットを、行2のデータがその時点でキャッシュにロックされることを指示する「1」にセットする。MRUビットとロックビットの「論理和」演算により作成される合成マスクは「1100」であり、キャッシュ行0及び1は依然として利用可能であることを指示する。ステップ4では、行2のデータについてヒットが起こる。キャッシュ制御装置はキャッシュメモリ行2に関わるMRUビットを「1」にセットさせて、行2のデータが近い時点で使用されたことを指示する。この合成マスクは「1100」のままであり、キャッシュ行0及び1は依然として利用可能であることがわかる。ステップ5では、行0にあるデータについてヒットが起こる。キャッシュ制御装置はキャッシュ

メモリ行0に関わるMRUビットを「1」にセットさせて、行0のデータが近い時点で使用されたことを指示する。この結果得られる合成マスクは「1101」であり、置換えのために利用可能のままであるのは行1のみであることがわかる。

【0029】ステップ6では、行1のデータについてヒットが起こる。キャッシュ制御装置がMRUビットを「1」にセットさせると、合成マスクは「1111」になってしまうであろう。その代わりに、キャッシュ制御装置はMRUビットをリセットさせ且つキャッシュメモリ行1に関わるMRUビットを「1」にセットさせて、行1のデータが近い時点で使用されたことを指示する。行2に関わるロックビットはセット状態のままであるので、その結果得られる合成マスクは「0110」になる。ステップ7では、ユーザープログラムは行3のデータをロックするための命令を実行する。キャッシュ制御装置は行3に関わるロックビットを「1」にセットさせることによりこの命令を実行する。ステップ8では、行0についてキャッシュヒットが起こる。「1111」の合成マスクが形成されるのを阻止するために、キャッシュ制御装置はMRUビットを再度クリアしなければならない。ステップ9では、キャッシュメモリ行1をロックする。そこで、ロックできる全てのキャッシュメモリ行がロックされたことになる。キャッシュメモリのデッドロックを阻止するため、システムはMRUビットをクリアする。キャッシュメモリ行1～3の全てがロックされたとき、置換えのために利用可能なキャッシュメモリ行は行0のみである。ステップ10では、行0についてヒットが起きる。合成マスクがキャッシュメモリのデッドロックを引起こす「1111」になってしまうので、行0に関わるMRUビットがキャッシュ制御装置によりセットされることはない。

【0030】ステップ11では、行1についてキャッシュヒットが起こる。行1に関わるMRUビットは、その行が近い時点で使用されたことを指示する「1」にセットされる。依然として、利用可能なキャッシュメモリ行は行0のみである。ステップ12では、行2のロックを解除することにより、キャッシュメモリ行2を最終的にロック解除する。そこで、合成マスクは「1010」になるので、置換えのために行0及び2を利用できるようになったことになる。ステップ13では、行0についてヒットが起こって、行0に関わるMRUビットは「1」にセットされる。ステップ10とは異なり、他の行はロック解除されているので、行0のMRUビットをセットしても、デッドロックは起こらない。

【0031】先に述べた通り、本発明のキャッシュシステムにおいてロッキングメカニズムを利用することにより得られるきわ立った利点は、キャッシュ置換えプロセスにさらに知能が追加されることである。ロックビットはアプリケーションプロセスによりセットされるので、キャッシュ制御装置レベルで知識を得ようとするために要求される知能は削除される。いくつかのキャッシュメモリ行をロックする要求を実行する方法の1つは、アプリケーションプログラムがそのような要求を所定の指令又はサブルーチン呼出しの形態でアプリケーションプログラムの中にプログラミングするというものである。プログラムの実行中に、あるいくつかの変数又はメモリ記憶場所を頻繁にアクセスすべきであるということがプログラマーにわかっていれば、第1回のアクセスの後、対応するロックビットをセットするために、特殊な指令を発行しても良い。このプログラムをコンパイルするコンパイラは指令要求を認識し、指令を実行するための適正なコードを提供する。

【0032】本発明の中で挙げるようなロッキングを制御するために、オペレーティングシステムルーチン、何らかのデータベース又はウィンドウシステムルーチンなどのシステムプログラムを使用しても良い。システムプログラムで実行されるロッキングは、アプリケーションプログラマーからの介入なく、アプリケーションプログラムが使用するいくつかのキーとなる機能の性能を向上させる。たとえば、図形パッケージを作成しているプログラマーであれば、オペレーティングシステムの図形ライブラリにより提供される効率良い線描出機能を使用するであろう。この機能をキャッシュにロックしたならば、図形パッケージの実行速度を間接的に増すことができる。

【0033】本発明のロッキングメカニズムは、スーパーバイザモードのみで実行するために利用可能な特殊アセンブリ言語命令を介して使用するように設けられているのが好ましい。プログラマーを補助するために、行ロック指令及び行ロック解除指令を提示するシステム呼出しを容易に書込むことができる。これは非常に強力なメカニズムであり、知識をもつプログラマーのみがこのメカニズムを使用すべきである。たとえば、SPARC™（SPARCはSPARC International, Inc. の商標である）アーキテクチャでは、ロックビットを変更するためにロード／記憶命令を適合させることが可能である。ロード／記憶指令を適合させる方法の1つは、ASI値をロックビットの場所に対応するように予約するというものである。CPUがその命令を実行するとき、キャッシュ制御装置はいくつかのロックビットをロック解除／ロックするためにCPUから指令を受けとる。キャッシュ制御装置はタグアレイ中の指定ロックビットをセット／リセットする指令を発行することによって応答する。ロード／記憶命令の詳細については、「The SPARC Architecture Manual」第8版の45～49ページ（Prentiss Hall, 1992年刊）を参照。

【0034】あるいは、対応するロックビットをセットさせることにより利益を得ると考えられる頻度の高いメ

13

モリアクセスを確定するために、実行すべきメモリアクセスについて自動化解析を実行するインテリジェントコンパイラを設けるのが好ましい。そのようにすれば、ロックビットのロッキング、続いてロック解除を実行するように、コンパイル済コードに自動的に指令を挿入することができる。この技法によれば、キャッシュであるいくつかのアクセスをロックすべきか否かの決定がコンパイラにより自動的に下され、アプリケーションプログラマーをそのような決定の実行から解放すると考えられるので、有利である。

【００３５】以上説明したようなロッキング機能を伴うＰＬＲＵを実現するキャッシュシステムでは、普通のＰＬＲＵキャッシュシステムと比べて、キャッシュメモリミスの率は著しく低い。そのように高い効率が得られるのは、キャッシュ置換えヒューリスティックに「知能」が追加されているためである。以上、ロッキング機能を伴う疑似ＬＲＵ置換え方式によってキャッシュメモリシステムを実現する方法及び装置を説明した。本発明の素子の材料及び配置について、本発明の趣旨から逸脱せずに当業者により変更及び変形を実施しうると考えられる。

【図面の簡単な説明】

14

【図１】従来の典型的なキャッシュメモリシステムの高レベルブロック線図。

【図２】従来の疑似最低使用頻度置換えプロセスの１例を示す図。

【図３】従来のセットアソシエイティブキャッシュを示す図。

【図４】本発明のキャッシュシステムの好ましい一実施例及び採用するロッキングビットを示す図。

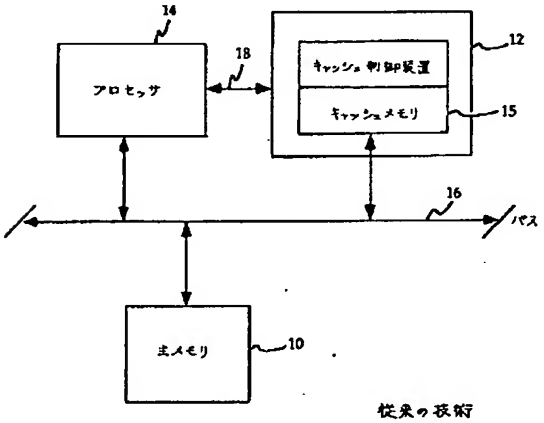
【図５】本発明のキャッシュシステムの好ましい実施例で利用するＳＴＡＧテーブル及びＰＴＡＧテーブルを示す図。

【図６】ロッキングビットを採用する疑似最低使用頻度置換えプロセスを示す図。

【符号の説明】

３００　キャッシュ
３１０　キャッシュ制御装置
３１５　タグテーブル
３２５　アドレス
３３０　制御ビット
３３５　ＭＲＵ
３４０　ロックビット

【図１】



従来の技術

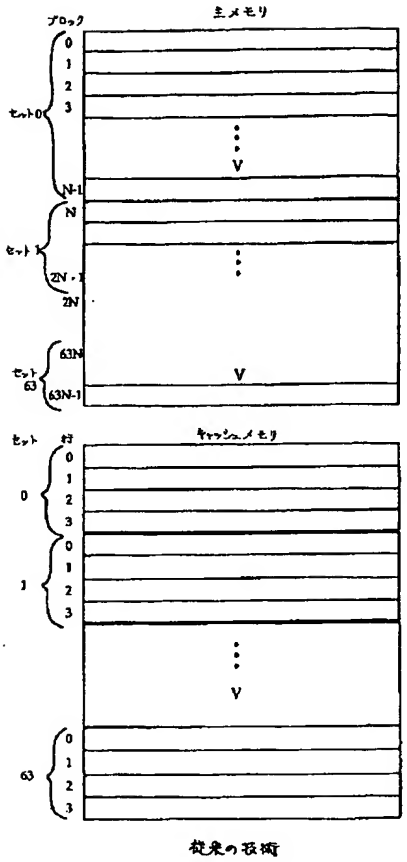
【図２】

PLRU

ステップ	アクション	MRUビット 3210	状態
1	初期状態	0000	行0,1,2及び3は利用可能
2	行3をアクセス	1000	行0,1及び2は利用可能
3	行1をアクセス	1010	行0及び2は利用可能
4	行0をアクセス	1011	行2は利用可能
5	行2をアクセス	0100	行0,1及び3は利用可能；最長喪失
6	行3をアクセス	1100	行0及び1は利用可能

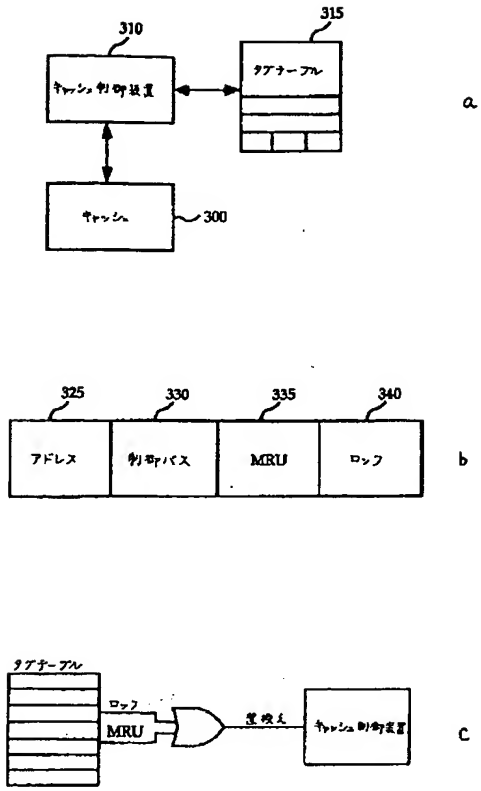
（従来の技術）

【図3】

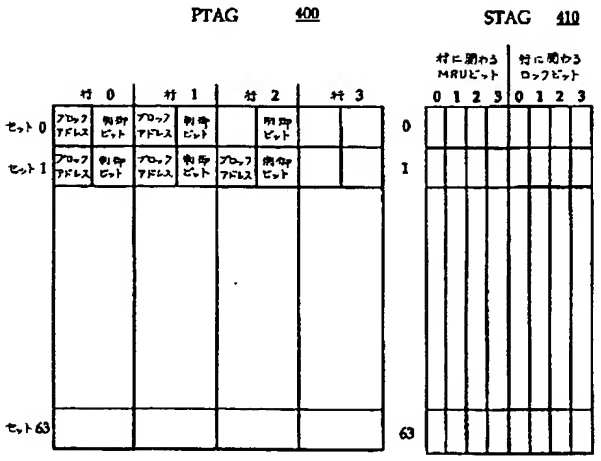


従来の技術

【図4】



【図5】



【図6】

PLRU＋ロッキング

ステップ	アクション	置換えビット 3210	ロックビット 3210	マスク 3210	状態
1		0000	0000	0000	初期条件
2	行3をアクセス	10000	0000	1000	行0，1及び2は利用可能
3	行2をロック	1000	0100	1100	行0及び1は利用可能
4	行2をアクセス	1100	0100	1100	行0及び1は利用可能
5	行0をアクセス	1101	0100	1101	行1は利用可能
6	行1をアクセス	0010	0100	0110	MRUビットをクリア，行3及び0は利用可能
7	行3をロック	0010	1100	1110	行0は利用可能
8	行0をアクセス	0001	1100	1101	MRUビットをクリア，行1は利用可能
9	行1をロック	0000	1110	1110	MRUビットをクリア，行0は利用可能
10	行0をアクセス	0000	1110	1110	行0は利用可能
11	行1をアクセス	0010	1110	1110	行0は利用可能
12	行2をロック解除	0010	1010	1010	行0及び2は利用可能
13	行0をアクセス	0011	1010	1011	行2は利用可能

フロントページの続き

(72)発明者　ラジヴ・エヌ・パテル
アメリカ合衆国　95148　カリフォルニア州・サン　ホゼ・ホワイトサンド　ドライブ・3116

(72)発明者　ノーマン・エム・ヘイーズ
アメリカ合衆国　94087　カリフォルニア州・サニーヴェイル・メリマック　ドライブ・1121